Домашние задания

Домашнее задание 1. Установка и использование СУБД

1. Установите систему управления реляционными базами данных.
2. Узнайте, как в вашей СУБД исполнять SQL в интерактивном режиме.
3. Узнайте, как в вашей СУБД исполнять SQL в пакетном режиме.
4. Разберитесь, как в вашей СУБД осуществляется поддержка русского языка.
5. Создайте базу данных и наполните ее в соответствии с примерами из презентации.

Ожидаемая структура проекта

1. Текстовая часть
    1. Описание предметной области с кратким описанием неочевидных сущностей и атрибутов.
    2. Предварительное разбиение на отношения (может отсутствовать).
    3. Для каждого отношения: определение функциональных зависимостей, нормализация до 5НФ денормализация (при необходимости).
    4. Модель сущность-связь.
    5. Физическая модель (должна соответствовать ERM) с указанием типов для доменов.
2. Часть на SQL
    - `ddl.sql` – описание таблиц и индексов.
    - `data.sql` – добавление тестовых данных.
    - `selects.sql` – запросы на получение данных и представления.
    - `updates.sql` – запросы на изменение данных, хранимые процедуры и триггеры.

Домашнее задание 2. Моделирование БД «Университет»

Спроектируйте базу данных «Университет», позволяющую хранить информацию о студентах, группах, преподавателях, дисциплинах и оценках. Поддержка дисциплин по выбору не требуется.

1. Составьте модель сущность-связь.
2. Преобразуйте модель сущность-связь в физическую модель.
3. Запишите физическую модель на языке SQL. Запись должна включать объявления ограничений.
4. Создайте базу данных по спроектированной модели.
5. Запишите операторы SQL, заполняющие базу тестовыми данными.

Форма для сдачи ДЗ

В рамках проекта:

1. Выберите тему проекта.
2. Сделайте предварительную схему для БД проекта на основе моделей.
3. Форма для тем проектов

Домашнее задание 3. Функциональные зависимости в БД «Университет»

Дано отношение с атрибутами *StudentId*, *StudentName*, *GroupId*, *GroupName*, *CourseId*, *CourseName*, *LecturerId*, *LecturerName*, *Mark*.

1. Найдите функциональные зависимости в данном отношении.
2. Найдите все ключи данного отношения.
3. Найдите замыкание множеств атрибутов:
    1. *GroupId*, *CourseId*;
    2. *StudentId*, *CourseId*;
    3. *StudentId*, *LecturerId*.
4. Найдите неприводимое множество функциональных зависимостей для данного отношения.

Форма для сдачи ДЗ

В рамках проекта:

1. Определите набор атрибутов, необходимых для проекта, и определите отношения на них.
2. Найдите функциональные зависимости полученных отношений.
3. Найдите все ключи полученных отношений.
4. Найдите неприводимые множества функциональных зависимостей для полученных отношений.

Домашнее задание 4. Нормализация БД «Университет»

Дано отношение с атрибутами *StudentId*, *StudentName*, *GroupId*, *GroupName*, *CourseId*, *CourseName*, *LecturerId*, *LecturerName*, *Mark*.

1. Инкрементально приведите данное отношение в пятую нормальную форму.
2. Постройте соответствующую модель сущность-связь.
3. Постройте соответствующую физическую модель.
4. Реализуйте SQL-скрипты, создающие схему базы данных.
5. Создайте базу данных по спроектированной модели.
6. Заполните базу тестовыми данными.

Форма для сдачи ДЗ

В рамках проекта:

1. Приведите схему базы в пятую нормальную форму.
2. Если итоговая схема не будет в НФ-5, то обоснуйте принятое решение.
3. Запишите определения таблиц на языке SQL.
4. Запишите на языке SQL наполнение таблиц тестовым данными.

Домашнее задание 5. Реляционная алгебра

Структура базы данных «Университет»:

- *Students(StudentId, StudentName, GroupId)*
- *Groups(GroupId, GroupName)*

- *Courses(CourseId, CourseName)*
- *Lecturers(LecturerId, LecturerName)*
- *Plan(GroupId, CourseId, LecturerId)*
- *Marks(StudentId, CourseId, Mark)*

Составьте выражения реляционной алгебры и соответствующие SQL-запросы, позволяющие получать

1. Информацию о студентах
    1. С заданным идентификатором (*StudentId*, *StudentName*, *GroupId* по *:StudentId*).
    2. С заданным ФИО (*StudentId*, *StudentName*, *GroupId* по *:StudentName*).
2. Полную информацию о студентах
    1. С заданным идентификатором (*StudentId*, *StudentName*, *GroupName* по *:StudentId*).
    2. С заданным ФИО (*StudentId*, *StudentName*, *GroupName* по *:StudentName*).
3. Информацию о студентах с заданной оценкой по дисциплине
    1. С заданным идентификатором (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:CourseId*).
    2. С заданным названием (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:CourseName*).
    3. Которую у него вёл лектор заданный идентификатором (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:LecturerId*).
    4. Которую у него вёл лектор, заданный ФИО (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:LecturerName*).
    5. Которую вёл лектор, заданный идентификатором (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:LecturerId*).
    6. Которую вёл лектор, заданный ФИО (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:LecturerName*).
4. Информацию о студентах не имеющих оценки по дисциплине
    1. Среди всех студентов (*StudentId*, *StudentName*, *GroupId* по *:CourseName*).
    2. Среди студентов, у которых есть эта дисциплина (*StudentId*, *StudentName*, *GroupId* по *:CourseName*).
5. Для каждого студента ФИО и названия дисциплин
    1. Которые у него есть по плану (*StudentName*, *CourseName*).
    2. Есть, но у него нет оценки (*StudentName*, *CourseName*).
    3. Есть, но у него не 4 или 5 (*StudentName*, *CourseName*).
6. Идентификаторы студентов по преподавателю
    1. Имеющих хотя бы одну оценку у преподавателя (*StudentId* по *:LecturerName*).
    2. Не имеющих ни одной оценки у преподавателя (*StudentId* по *:LecturerName*).
    3. Имеющих оценки по всем дисциплинам преподавателя (*StudentId* по *:LecturerName*).
    4. Имеющих оценки по всем дисциплинам преподавателя, которые он вёл у этого студента (*StudentId* по *:LecturerName*).
7. Группы и дисциплины, такие что все студенты группы сдали эту дисциплину
    1. Идентификаторы (*GroupId*, *CourseId*).
    2. Названия (*GroupName*, *CourseName*).

Составьте SQL-запросы, позволяющие получать

8. Суммарный балл
    1. Одного студента (*SumMark* по *:StudentId*).
    2. Каждого студента (*StudentName*, *SumMark*).
    3. Каждой группы (*GroupName*, *SumMark*).
9. Средний балл
    1. Одного студента (*AvgMark* по *:StudentId*).
    2. Каждого студента (*StudentName*, *AvgMark*).
    3. Каждой группы (*GroupName*, *AvgMark*).
    4. Средний балл средних баллов студентов каждой группы (*GroupName*, *AvgAvgMark*).
10. Для каждого студента: число дисциплин, которые у него были, число сданных дисциплин и число несданных дисциплин (*StudentId*, *Total*, *Passed*, *Failed*).

Тестовый полигон

Технические особенности проверки.

- Сдача — в PCMS.
- Проверяться и оцениваться будет **последняя** посланная версия.
- Проверка разделена на 4 фазы:
    1. пустые таблицы (синтаксис и набор столбцов);
    2. таблицы с не более чем одной записью;
    3. таблицы с простыми данными;
    4. таблицы со сложными данными.
- В случае проблем с синтаксисом или набором столбцов вы будете получать *Presentaion Error*.
- Реляционная алгебра проверяется одним тестом на фазу, движком из тестового полигона.
- SQL проверяется тремя тестами на фазу — с разными СУБД. Первая СУБД — SQLite, как на тестовом полигоне.
- Известные спецэффекты:
    - SQLite поддерживает только `left join`. `right`/`outer join` делаются через него.
    - Все вложенные запросы надо именовать, даже если вы не будете использовать это имя:

        ```
        select ... from ... (select ... ) SubQueryName ...
        ```

    - Используйте данные из минимально возможного набора таблиц.

Домашнее задание 6. Реляционное исчисление

Составьте запросы в терминах языков Datalog и SQL для базы данных «Университет», позволяющие получать:

1. Информацию о студентах
    1. С заданным ФИО (*StudentId*, *StudentName*, *GroupId* по *:StudentName*).
    2. Учащихся в заданной группе (*StudentId*, *StudentName*, *GroupId* по *:GroupName*).
    3. С заданной оценкой по дисциплине, заданной идентификатором (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:CourseId*).
    4. С заданной оценкой по дисциплине, заданной названием (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:CourseName*).
2. Полную информацию о студентах

    1. Для всех студентов (*StudentId*, *StudentName*, *GroupName*).
    2. Студентов, не имеющих оценки по дисциплине, заданной идентификатором (*StudentId*, *StudentName*, *GroupName* по *:CourseId*).
    3. Студентов, не имеющих оценки по дисциплине, заданной названием (*StudentId*, *StudentName*, *GroupName* по *:CourseName*).
    4. Студентов, не имеющих оценки по дисциплине, у которых есть эта дисциплина (*StudentId*, *StudentName*, *GroupName* по *:CourseId*).
    5. Студентов, не имеющих оценки по дисциплине, у которых есть эта дисциплина (*StudentId*, *StudentName*, *GroupName* по *:CourseName*).
3. Студенты и дисциплины, такие что у студента была дисциплина (по плану или есть оценка)
    1. Идентификаторы (*StudentId*, *CourseId*).
    2. Имя и название (*StudentName*, *CourseName*).
4. Студенты и дисциплины, такие что дисциплина есть в его плане и у студента долг по этой дисциплине
    1. Долгом считается отсутствие оценки (*StudentName*, *CourseName*).
    2. Долгом считается оценка не выше 2 (*StudentName*, *CourseName*).
    3. Долгом считается и отсутствие оценки и оценка не выше 2 (*StudentName*, *CourseName*).
5. Идентификаторы студентов по преподавателю
    1. Имеющих хотя бы одну оценку у преподавателя (*StudentId* по *:LecturerName*).
    2. Не имеющих ни одной оценки у преподавателя (*StudentId* по *:LecturerName*).
    3. Имеющих оценки по всем дисциплинам преподавателя (*StudentId* по *:LecturerName*).
    4. Имеющих оценки по всем дисциплинам преподавателя, которые он вёл у этого студента (*StudentId* по *:LecturerName*).
6. Группы и дисциплины, такие что все студенты группы сдали дисциплину
    1. Идентификаторы (*GroupId*, *CourseId*).
    2. Названия (*GroupName*, *CourseName*).

Примечания

1. В Datalog итоговым считается последнее объявленное отношение.
2. Текущая реализация Datalog не поддерживает рекурсивные определения.
3. В SQL-запросах нельзя использовать `* join`.

В рамках проекта:

1. Определите запросы (в том числе, агрегирующие), необходимые для работы проекта.
2. Реализуйте запросы на языке SQL.

Домашнее задание 7. Изменение данных

Реализуйте указанные запросы, представления, проверки и триггеры на языке SQL.

1. Напишите запросы, удаляющие студентов
    1. Учащихся в группе, заданной идентификатором (*GroupId*).
    2. Учащихся в группе, заданной названием (*GroupName*).
    3. Без оценок.

    4. Имеющих 3 и более оценки.
    5. Имеющих 3 и менее оценки.
    6. Студентов, с долгами (здесь и далее — по отсутствию оценки).
    7. Студентов, имеющих 2 и более долга.
    8. Студентов, имеющих не более 2 долгов.
2. Напишите запросы, обновляющие данные студентов
    1. Изменение имени студента (*StudentId*, *StudentName*).
    2. Перевод студента из группы в группу по индентификаторам (*StudentId*, *GroupId*, *FromGroupId*).
    3. Перевод всех студентов из группы в группу по идентификаторам (*GroupId*, *FromGroupId*).
    4. Перевод студента из группы в группу по названиям (*GroupName*, *FromGroupName*).
    5. Перевод всех студентов из группы в группу, только если целевая группа существует (*GroupName*, *FromGroupName*).
3. Напишите запросы, подсчитывающие статистику по студентам
    1. Число оценок студента (столбец *Students.Marks*) (*StudentId*).
    2. Число оценок каждого студента (столбец *Students.Marks*).
    3. Пересчет числа оценок каждого студента по данным из таблицы *NewMarks* (столбец *Students.Marks*).
    4. Число сданных дисциплин каждого студента (столбец *Students.Marks*).
    5. Число долгов студента (столбец *Students.Debts*) (*StudentId*).
    6. Число долгов каждого студента (столбец *Students.Debts*).
    7. Число долгов каждого студента группы (столбец *Students.Debts*) (*GroupName*).
    8. Число оценок и долгов каждого студента (столбцы *Students.Marks*, *Students.Debts*).
4. Напишите запросы, обновляющие оценки, с учетом данных из таблицы *NewMarks*, имеющей такую же структуру, как таблица *Marks*
    1. Проставляющий новую оценку только если ранее оценки не было.
    2. Проставляющий новую оценку только если ранее оценка была.
    3. Проставляющий максимум из старой и новой оценки только если ранее оценка была.
    4. Проставляющий максимум из старой и новой оценки (если ранее оценки не было, то новую оценку).
5. Работа с представлениями
    1. Создайте представление *StudentMarks* в котором для каждого студента указано число оценок (*StudentId*, *Marks*).
    2. Создайте представление *AllMarks* в котором для каждого студента указано число оценок, включая оценки из таблицы *NewMarks* (*StudentId*, *Marks*).
    3. Создайте представление *Debts* в котором для каждого студента, имеющего долги указано их число (*StudentId*, *Debts*).
    4. Создайте представление *StudentDebts* в котором для каждого студента указано число долгов (*StudentId*, *Debts*).

6. Целостность данных.

Обратите внимание, что задания из этого раздела надо посылать в PCMS, но они будут проверяться только вручную после окончания сдачи. То есть в PCMS вы получите + за любое решение.

В комментарии перед запросом укажите версию использованной СУБД.

    1. Добавьте проверку того, что у студентов есть оценки только по дисциплинам из их плана (*NoExtraMarks*) (*StudentId*, *CourseId*).
    2. Добавьте проверку того, что все студенты каждой группы имею оценку по одному и тому же набору дисциплин (*SameMarks*). (*StudentId*).
    3. Создайте триггер *PreserveMarks*, не позволяющий уменьшить оценку студента по дисциплине. При попытке такого изменения оценка изменяться не должна. (*StudentId*).

1. Напишите запросы, удаляющие студентов:
    1. Учащихся в группе *:GroupId*;
    2. Учащихся в группе *:GroupName*;
    3. Без оценок;
    4. Имеющих 3 и более оценки;
    5. Имеющих 3 и менее оценки;
    6. Студентов, с долгами (здесь и далее — по отсутствию оценки);
    7. Студентов, имеющих 2 и более долга;
    8. Студентов, имеющих не более 2 долгов.
2. Напишите запросы, обновляющие данные студентов:
    1. Изменение имени студента *:StudentId* на *:StudentName*;
    2. Перевод студента *:StudentId* из группы *:FromGroupId* в группу *:GroupId*;
    3. Перевод всех студентов из группы *:FromGroupId* в группу *:GroupId*;
    4. Перевод всех студентов из группы *:FromGroupName* в группу *:GroupName*;
    5. Перевод всех студентов из группы *:FromGroupName* в группу *:GroupName* только если целевая группа существует;
3. Напишите запросы, подсчитывающие статистику по студентам:
    1. Число оценок студента *:StudentId* (столбец *Marks*);
    2. Число оценок каждого студента (столбец *Marks*);
    3. Пересчет числа оценок каждого студента по данным из таблицы *NewMarks* (столбец *Marks*);
    4. Число сданных дисциплин каждого студента (столбец *Marks*);
    5. Число долгов студента *:StudentId* (столбец *Debts*);
    6. Число долгов каждого студента (столбец *Debts*);
    7. Число долгов каждого студента группы *:GroupName* (столбец *Debts*);
    8. Число оценок и долгов каждого студента (столбцы *Marks*, *Debts*);
4. Напишите запросы, обновляющие оценки, с учетом данных из таблицы *NewMarks*.
    1. Проставляющий новую оценку только если ранее оценки не было.
    2. Проставляющий новую оценку только если ранее оценка была.
    3. Проставляющий максимум из старой и новой оценки только если ранее оценка была.
    4. Проставляющий максимум из старой и новой оценки (если ранее оценки не было, то новую оценку).
5. Работа с представлениями
    1. Создайте представление *StudentMarks* в котором для каждого студента указано число оценок (столбцы *StudentId*, *Marks*);
    2. Создайте представление *AllMarks* в котором для каждого студента указано число оценок, включая оценки из таблицы *NewMarks* (столбцы *StudentId*, *Marks*);
    3. Создайте представление *Debts* в котором для каждого студента, имеющего долги указано их число (столбцы *StudentId*, *Debts*);

4. Создайте представление *StudentDebts* в котором для каждого студента указано число долгов (столбцы *StudentId*, *Debts*);

6. Целостность данных.

   Обратите внимание, что задания из этого раздела надо посылать в PCMS, но они будут проверяться только вручную после окончания сдачи. То есть в PCMS вы получите + за любое решение.

   В комментарии перед запросом укажите версию использованной СУБД.

   1. Добавьте проверку того, что у студентов есть оценки только по дисциплинам из их плана (*NoExtraMarks*).
   2. Добавьте проверку того, что все студенты каждой группы имею оценку по одному и тому же набору дисциплин (*SameMarks*).
   3. Создайте триггер *PreserveMarks*, не позволяющий уменьшить оценку студента по дисциплине. При попытке такого изменения оценка изменяться не должна.

В рамках проекта:

1. Определите модифицирующие запросы, необходимые для работоспособности проекта.
2. Запишите эти запросы на языке SQL.

Домашнее задание 8. Индексирование

1. Определите, какие индексы требуется добавить к таблицам базы данных Университет» на основе запросов из ДЗ-5, 6 и 7.
2. Пусть частым запросом является определение среднего балла студентов группы по дисциплине. Как будет выглядеть запрос и какие индексы могут помочь при его исполнении?
3. Придумайте три запроса, требующих новых индексов и запишите их. Если в результате, некоторые из старых индексов станут бесполезными, удалите их.

При выполнении задания считайте, что ФЗ соответствуют полученным в ДЗ-3 и 4.

Форма для сдачи ДЗ

В рамках проекта:

1. Определите индексы (и их типы), необходимые для эффективного исполнения запросов.
2. Запишите определения индексов на языке SQL.

Домашнее задание 9. Хранимые процедуры

В базе данных `Airline` информация о рейсах самолётов задана в виде таблиц

```
Flights(
    FlightId integer,
    FlightTime timestamp,
    PlaneId integer,
    -- Дополнительные столбцы, при необходимости
```

```
)
Seats(
    PlaneId integer,
    SeatNo varchar(4), -- 123A
    -- Дополнительные столбцы, при необходимости
)
```

Реализуйте запросы к базе данных `Airline` с применением представлений, хранимых процедур и функций. При необходимости, вы можете создать дополнительные таблицы, представления и хранимые процедуры.

1. `FreeSeats(FlightId)` — список мест, доступных для продажи и для бронирования.
2. `Reserve(UserId, Pass, FlightId, SeatNo)` — пытается забронировать место на трое суток начиная с момента бронирования. Возвращает *истину*, если удалось и *ложь* — в противном случае.
3. `ExtendReservation(UserId, Pass, FlightId, SeatNo)` — пытается продлить бронь места на трое суток начиная с момента продления. Возвращает *истину*, если удалось и *ложь* — в противном случае.
4. `BuyFree(FlightId, SeatNo)` — пытается купить свободное место. Возвращает *истину*, если удалось и *ложь* — в противном случае.
5. `BuyReserved(UserId, Pass, FlightId, SeatNo)` — пытается выкупить забронированное место (пользователи должны совпадать). Возвращает *истину*, если удалось и *ложь* — в противном случае.
6. `FlightsStatistics(UserId, Pass)` — статистика по рейсам: возможность бронирования и покупки, число свободных, забронированных и проданных мест.
7. `FlightStat(UserId, Pass, FlightId)` — статистика по рейсу: возможность бронирования и покупки, число свободных, забронированных и проданных мест.
8. `CompressSeats(FlightId)` — оптимизирует занятость мест в самолете. В результате оптимизации, в начале самолета должны быть купленные места, затем — забронированные, а в конце — свободные. Примечание: клиенты, которые уже выкупили билеты также должны быть пересажены.

Форма для сдачи ДЗ

В рамках проекта:

1. Определите храниемые процедуры и функции, необходимые для работы проекта.
2. Реализуйте хранимые процедуры (функций) на языке SQL.

Домашнее задание 10. Транзакции

Спланируйте транзакции и выберите их уровни изоляции для базы данных `Airline`.

1. Для каждой хранимой процедуры из предыдущего домашнего задания выберите минимальный допустимый уровень изоляции транзакций (с обоснованием).
2. Реализуйте сценарий работы:
    1. Запрос списка свободных мест.
    2. Отображение списка свободных мест пользователю.
    3. Бронирование или покупка места, выбранного пользователем.

Форма для сдачи ДЗ

В рамках проекта:

1. Определите минимальный уровень изоляции транзакций, необходимый для каждого запроса и хранимой процедуры.

Домашнее задание 11. Онлайн-активности

Виды активностей:

- разбор домашнего задания (очная);
- разбор новой темы (очная);
- разметка видео;
- написание wiki-конспекта.

Можно участвовать только в одной активности. Задания распределяются в порядке записи. Форма для записи.

Разметка видео

- Размечаются оба видео за неделю.
- Лекция:
  - должны быть обозначены (под)разделы, выделенные в презентации;
  - (под)разделы длиннее 6 минут должны быть разбиты на логические фрагменты.
- Разбор новой темы:
  - должно быть обозначено каждое задание;
  - (под)задания длиннее 6 минут должны быть разбиты на логические фрагменты.
- Разбор домашнего задания:
  - пункты задания короче 2 минут можно объединять в логические блоки;
  - пункты задания длиннее 6 минут должны быть разбиты на логические фрагменты.

Конспекты

- Конспект должен быть разбит на страницы в соответствии с разделами презентации.
- На странице должны быть выделены подразделы, соответствующие презентации (если такие есть).
- Можно и нужно использовать картинки и примеры из презентаций.
- Вместо скриншотов лучше брать оригинальные svg-файлы.